render the claims obvious (as explained further below). However, since the present invention was made for particular advantage in the circumstances of the computer graphics shading task which requires a highly effective utilization of parallel processor resources, Claims 1-6, 8-18, and 20 are amended to specifically claim the invention subject matter as applied to the CG shading task where it has the most pronounced effect over conventional parallel processing methods. Claims 7 and 19 which were specifically directed to the shading task environment are canceled as now being redundant. This amendment of claims is submitted without prejudice to the right of the Applicant to later seek added patent coverage for application of the same principles in the invention to other types of computationally intensive tasks.

The amended claims are now specifically directed to the method of grouping shading task processes which are not dependent on the passing of control from any external process into a shading task space. The shading task space comprises a plurality of shading task objects which can be executed in one computational step once all of its "data-waiting" slots are filled with data passed from other relevant task objects in the shading task space. The shading task objects are scheduled for processing according to three states, wherein if they are awaiting data they are placed in a "waiting" state in which they are not assigned to any processor, wherein if they have all "data waiting" slots filled they are placed in an "active" state where they are assigned to an "unoccupied" processor for processing, and wherein once the computational step has been completed they are placed in a "dead" state (from which they can be deleted from the shading task space) and the processor can again be placed in an "unoccupied" state.

The Shapiro patent cited by the Examiner appears to provide only a general discussion of methods for constructing an executable model for a complex system using successive hierarchies of processing tasks. For example, its Figure 10 appears to illustrate an "assembly line" or "pipeline" representation of the handling of the hierarchies of processing tasks. The Request 204 activates Machine 1 which performs its task and stores the result in a Buffer 1, from which the data goes to Machine 2 which performs its task and stores the result in a Buffer 2, from which the data goes to Machine 3 which performs its task and the final result is passed on to a next or new processing stage. Nowhere does the Shapiro reference disclose placing task objects requiring only its "data waiting" slots to be filled in a common pool of task objects waiting to be processed, so as to avoid assigning a task object to a processor while its "data waiting" slots remain to be filled. The present invention seeks to avoid the problems in prior "assembly line" or "pipeline" models of parallel processing where a processor that takes longer to complete its task than others or has a task that is awaiting other data in order to be completed can keep the other processors in the pipeline waiting.

The Boland patent cited by the Examiner appears to take a different approach than the present invention. In Boland, a task process awaiting data is placed in a "sleep" state while awaiting completion of another task process that will supply the data before it can be placed on "wakeup" status for processing. An affinity scheduling system is used to detect pairs of processes that frequently exchange "wakeup" requests in order to <u>assign these "affinitized" processes to the same processor</u> (col. 4, lines 31-49) so that they can be prioritized to be performed in succession by the same processor. The present invention does not employ the Boland affinity scheduling, but rather keeps all grouped task objects awaiting data in a processing pool, and only when a task object has changed from "waiting" to "active" state is it assigned to the next "unoccupied" processor.

The IDF reference cited by the Examiner provides a general description of a visual programming environment for designing data flow models in computer graphics. This reference also does not disclose or suggest placing task objects requiring only its "data waiting" slots to be filled and capable of being processed in one computational step in a common pool of task objects waiting to be processed, so as to avoid assigning a task object to an unoccupied processor while its "data waiting" slots remain to be filled.

In summary, Claims 1-6, 8-18, and 20 are deemed to define subject matter that is clearly distinct from the cited prior art, considered singly or in combination. Allowance of these Claims is respectfully requested upon reconsideration.

This response is filed within the 3-month period allowed for response from the renewed mailing date of the action, and with numbers of claims within those paid for with the filing fee, so that no fees are deemed to be due for this amendment. However, if any fees are determined to be due for acceptance hereof, authorization is hereby given to charge our Deposit Account No. 501198 of the undersigned firm.

CERTIFICATE OF MAILING:

The undersigned certifies that the foregoing is being mailed on February 21, 2006, by depositing it with the U.S. Postal Service, first class postage paid, addressed to: Mail Stop: Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Respectfully submitted,
ATTORNEYS FOR APPLICANT

[illegible]

Leighton K. Chong
USPTO Reg. No. 27,621
GODBEY GRIFFITHS REISS & CHONG
1001 Bishop St., Pauahi Tower Suite 2300
Honolulu, HI 96813
Tel: (808) 523-8894
Fax: (808) 523-8899

includes a master shading task grouping of shading task spaces each of which performs shading of a pixel in the image frame.

(previously presented)

9. A parallel processing method according to Claim 8, wherein each shading task space includes a plurality of "pixel shading" task objects for performing shading of the pixel based upon ray shooting from light sources in the scene, and a "compositing" task object for compositing the shading results for the pixel.

(previously presented)

10. A parallel processing method according to Claim 9, wherein each shading task object has at least one "data-waiting" slot for return of data characterizing light emitted from a respective light source in the scene.

(previously presented)

11. A parallel processing method according to Claim 9, wherein the rendering task includes a function for receiving scene data for a "world map" of the scene, a function for defining the scene objects in each frame of the scene, a function for defining the pixels of an object in the scene intersected by an eye ray of a viewer of the scene, and a function for tiling together the shading results returned by each of the master shading task groupings for respective objects in the image frame.

(currently amended)

12. A software programming method for performing ~~processing~~ computer graphics shading tasks tasks in parallel on a plurality of processors comprising:

(a) identifying at least one shading task area of a large computer graphics rendering ~~processing~~ task directed to a plurality of ~~computational~~ computer shading task processes for shading an image frame of a scene that can be grouped together as a shading task space not dependent on passing of control of processing from an external process in order to complete processing of the computational processes of the shading task space;

(b) breaking down the shading task space into a plurality of self-contained shading